Mission Success with Software Development Principles: Application to Mars'98 Missions

Presented by:

Milton L. Lavin

Jet Propulsion Laboratory

November 30, 2000

Topics

- Rationale for these principles
- Development and use
- Content overview
- Application to Mars'98 Missions
  - Mars Climate Orbiter Navigation Software
  - Mars Polar Lander Touchdown Sensor
- Summary

Rationale for These Principles

- Internal study - flight software late and over budget
- Recent mission losses directly traceable to software process defects
- Continuing loss of experienced staff vs. explosion in the number of active missions
- Faster/better/cheaper is here to stay -- how to do it right?

Development and Use

- **Scope**
  - Initial focus on mission-critical software
  - Design and management over the life cycle
  - Both in-house and out-of-house development
- **Content & Style**
  - Make a difference in quality/cost/schedule at JPL
  - Organized into general and detailed principles
  - Succinct requirement, plus guidance
- **Sources**: Lessons Learned, Workshops, Loss Reports
- **Process**: Editorial Board/CCB, multiple drafts & reviews
- **Use:** Compliance in development plan and reviews

Content Overview

System Definition/System Engineering	13
Planning and Monitoring	17
Cost Estimation	4
Risk Management	3
Organization and Staffing	6
Design and Implementation	14
Integration and Test	16
Configuration Management	3
Software Acquisition	2
Product and Process Verification	4
Flight Software	20
Total	102

Illustrative Principle

Flight Software Design Principle:

4.15 “The software self-test and built-in test routines shall be removable for flight. If not removable, the test routines shall not cause flight hardware damage or interfere with the proper execution of the flight software if tests are inadvertently executed”

Application to Mars'98 Missions

Mars Climate Orbiter (MCO) Navigation Software:

- Used to infer spacecraft trajectory perturbations from angular momentum desaturation maneuvers
- Engineering units defect injected in requirements flowdown to Small Forces Software
- Software Principles (retrospectively) identify 10 opportunities to catch the error --
  - System engineering and design (2)
  - Walkthroughs of requirements, design, & code (3)
  - End-to-end testing by contractor and JPL (5)

Application to Mars'98 Missions

Mars Polar Lander (MPL) Touchdown Sensor Software:

- Used to command descent engine shutdown
- Defect introduced in flowdown of changed requirements to touchdown sensor software
- Software Principles (retrospectively) identify 16 opportunities to catch the error --
  - System engineering and design (5)
  - Walkthroughs of requirements, design, code (4)
  - End-to-end testing by contractor and JPL (7)

Applicability of Individual Principles

Software Development Principle	*MCO*	*MPL*
Planning and Monitoring		
3.2.7 Joint development planning for interfacing hardware and software		X
3.2.9 Identification of milestone and peer reviews	X	X
3.2.10 Participation of hardware engineers and operations team in reviews		X
3.2.12 Comprehensive peer review of intermediate products	X	X
Risk Management		
3.4.3 Early validation of interfaces, high-risk algorithms, and COTS	X	X
Design and Implementation		
3.6.3 Design traced to software and mission requirements	X	X
3.6.4 Documented analytical basis for logic design		X
3.6.6 Software logic to verify values of input and output parameters	X	X
Integration and Test		
3.7.6 Detailed testing of mission phase transitions		X
3.7.7 Testing to address Fault Tree Analysis and off-nominal hardware behavior		X
3.7.9 Stress testing used to aggressively find latent defects	X	X
3.7.13 Trace from final system test to mission requirements	X	X
Software Acquisition		
3.9.1 Project Implementation Plan to address management of software acquisition:		
• In-process JPL review of intermediate products	X	X
• JPL participation in pre-delivery testing	X	X
Product and Process Verification		
3.10.4 Acceptance test that exercises mission-critical systems	X	X
Flight Software		
4.5 Accommodation of nominal, off-nominal/transient inputs		X

Summary

- Defect injection during software creation not surprising, given its complexity and proneness to human error
- Injection of serious defects during requirements elaboration is not unusual (68%, according to GSFC)
- “Test as you Fly” encourages systematic application of software principles during development to
  - Deter defects
  - Identify and correct them prior to launch
- JPL’s software principles are being continually refined and elaborated using mission experience